ALPHA DATA

Building Linux for the ADMVPX39Z2

Introduction

This guide describes how to build the Open Source Embedded Linux distribution "Poky" using the Yocto build system, and "Das U-Boot", for an ADM-VPX3-9Z2.

Building Linux Kernel and Root File System

Required Tools

To follow this procedure you will require a PC or Virtual Machine running a Desktop Linux distribution. The system must be setup with the correct configuration before starting a build. See the 'The Linux Distribution' and 'The Build Host Packages' sections in http://www.yoctoproject.org for instructions on setting up the required packages on your Linux Desktop system.

Sources

After setting up a Linux desktop environment for the Yocto build system, **git** can be used to acquire the source code required. Start by creating a new directory for your Embedded Linux project. From inside the new directory use the following commands to acquire a copies of the required code bases:

```
git clone -b thud git://git.yoctoproject.org/poky.git
git clone -b thud https://github.com/openembedded/openembedded-core.git
git clone -b thud https://github.com/Xilinx/meta-xilinx.git
git clone -b thud https://github.com/adps/meta-admvpx39z.git
git clone -b admvpx39z2-thud https://github.com/adps/meta-adlnx.git
```

The last two code bases, **meta-adlnx** and **meta-admvpx39z**, are Alpha Data's example root files system, and Alpha Data's board support package for the ADM-VPX3-9Z2.

Building

From the **~/poky** sub-folder in your Embedded Linux project directory, enter the following command to initialise the build environment:

```
source oe-init-build-env
```

This will create a build directory, **build**, which will be switched into to as the active directory after the script completes execution.

Build setup

Before a build can be started, two files must be edited in the **~/poky/build/conf** directory; poky/build/conf/bblayers, and poky/build/conf/local.config

poky/build/conf/bblayers

The **bblayers** file must be modified to include the full path of additional layers that need to be added to the Yocto

Linux build system. Edit **bblayers** to be similar to the following, including the **meta-adlnx** and **meta-admxrc7z** and layers and the **openembedded-core/meta** layer. Note the path **/home/my_home/yocto_linux** should be changed to the full path of your Embedded Linux project directory.

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
/home/my_home/yocto_linux/openembedded-core/meta \
/home/my_home/yocto_linux/meta-xilinx \
/home/my_home/yocto_linux/meta-adlnx \
/home/my_home/yocto_linux/meta-admvpx39z \
/home/my_home/yocto_linux/poky/meta \
/home/my_home/yocto_linux/poky/meta-yocto \
/home/my_home/yocto_linux/poky/meta-yocto-bsp \
  "
```

poky/build/conf/local.config

The **local.config** file must be modified to specify the target machine. Edit this file to include the following lines to select the target machine and source mirror URL:

```
MACHINE ??= "admvpx39z"
```

Build

Use the following command in the **~/poky/build directory** to start a build of the Embedded Linux Kernel and root file system. This will take some time to complete.

```
bitbake adlnx-image
```

The output should look similar to the following:

```
Loading cache: 100% |################################################################
##########################################################################################
########################################################| Time: 0:00:03
Loaded 2558 entries from dependency cache.
Parsing recipes: 100% |##############################################################
##########################################################################################
########################################################| Time: 0:00:01
Parsing of 1610 .bb files complete (1609 cached, 1 parsed). 2559 targets, 126 skip
ped, 0 masked, 0 errors.
NOTE: Resolving any missing task queue dependencies

Build Configuration:
BB_VERSION           = "1.40.0"
BUILD_SYS            = "x86_64-linux"
NATIVELSBSTRING      = "ubuntu-14.04"
TARGET_SYS           = "aarch64-poky-linux"
MACHINE              = "admvpx39z"
DISTRO               = "poky"
DISTRO_VERSION       = "2.6.1"
TUNE_FEATURES        = "aarch64"
TARGET_FPU           = ""
meta                 = "thud:ad0a553f0bbdbed5f78a27162289a1e358580dcc"
meta-xilinx-bsp      = "rel-v2018.3:7922f16dfa5308fb5419a80f513bb07c0384f95e"
```

```
meta-adlnx            = "admvpx39z2:77f2f4531a95be779af11cc0ff2d04a5c0add69f"
meta-admvpx39z        = "master:3973f82b278aea6ed4df2aa84397fafa29f9c6f8"
meta
meta-poky
meta-yocto-bsp        = "thud:cc73390a75d98b96eb861ae0624283c1ea6ef1bd"



NOTE: Preparing runqueue
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
```

Output files

After the build is completed, several output files are created in **poky/build/tmp/deploy/images/admvpx39z/**

Image-admvpx39z.bin

The Linux Yocto Poky Thud kernel image. This is the core of the operating system, but requires a device tree to provide details of the hardware configuration.

admvpx39z.dtb

Device tree blob. This provides details of the hardware configuration to the kernel.

adlnx-image-admvpx39z.cpio.gz.u-boot

The root file system as a RAM disk containing a CPIO image prepended with a Das U-Boot header.

Building a standalone toolchain

In addition to building the root file system and Kernel a standalone toolchain might be needed for developing application outside the Yocto enviroment. This can be build with the following command.

```
bitbake adlnx-image -c populate_sdk
```

This will create the a toolchain installer script **poky/build/tmp/deploy/sdk/poky-glibc-x86_64-adlnx-image-aarch64-toolchain-2.2.4.sh**. Executing this on a Linux system will allow the user to install cross compiler tools customised to target the adlnx reference image.

After running the toolchain installer script, the environment can be setup for cross-compiling by running

```
source /opt/poky/2.2.4/environment-setup-aarch64-poky-linux
```

After this, the cross-compipler can be called using the environment variabls set by the environment setup script. e.g. gcc can be called with $CC, or g++ can be called with $CXX.

Das U-Boot

To make it easier to develop and update Embedded Linux images, the FSBL (First Stage Boot Loader) will load Das U-Boot. Das U-Boot is a second stage boot loader that allows a number of boot options, including booting from QSPI, SD card, and TFTP.

Sources

Acquire the sources to build Das U-Boot using the following command:

```
git clone -b rel-v2018.3 https://github.com/adps/u-boot-ad-zynqmp.git
```

Building Das U-Boot

Execute the following commands inside the u-boot-ad-zynqmp directory:

To set up the system for cross-compiling, source the environment setup script that was generated when building the standalone toolchain.

```
source /opt/poky/2.2.4/environment-setup-aarch64-poky-linux
```

```
make alphadata_zynqmp_admvpx39z_defconfig
```

```
make all
```

Output files

After building has completed, the following files can be found in the build directory:

u-boot.elf This file is an ELF version of the built U-Boot binary.

The u-boot.elf file can be used with Xilinx SDK's "bootgen" utility to create the BOOT.bin file required for booting. See https://github.com/adps/fsbl-vivado_admvpx39z2/blob/rel-v2018.3/doc/ad-ug_v1_1_fsbl_and_vivado_for_9Z2.pdf for infomration about building the FSBL and BOOT.bin files.

Testing

The default environment variables in Das U-Boot will attempt to boot the **admvpx39z2.bin** kernel image with the **adlnx-image-admvpx39z.cpio.gz.u-boot** root file system and **admvpx39z.dtb** device tree.

Preparing an SDCard

To test the built First Stage Bootloader, Bitstream, Das U-Boot and Embedded Linux image, follow this procedure:

1. Prepare a micro SD card with a FAT32 primary active partition as the first partition.
2. Copy the following files into FAT32 partition of the micro SD card: Image-admvpx39z.bin, adlnx-image-admvpx39z.cpio.gz.u-boot, admvpx39z.dtb, BOOT.bin
3. Configure the ADM-VPX3-9Z2 for boot from uSD, by setting SW4-1 to OFF, SW4-2 to OFF and SW4-3 to ON.
4. With the ADM-VPX3-9Z2 powered down, install the uSD card in the uSD card slot.
5. Connect a serial cable to the serial output from the ADM-VPX3-9Z2-RTM, and open a serial terminal at 115.2k.
6. After powering on the ADM-VPX3-9Z2 it should boot from the uSD card, first showing information about the ADM-VPX3-9Z2 in the FSBL, before proceeding to load and execute U-Boot. U-Boot will wait for three seconds for a user to intervene before starting to load Embedded Linux.

After Linux has booted, the user name **root** can be used to log into the system (without a password).

Running QEMU

Xilinx's QEMU tree should be built before trying to boot the admvpx39z2 Linux image on QEMU. Instructions for building QEMU can be found here:https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/18842060/QEMU

After building Xilinx-QEMU, change directory to meta-adlnx/qemu, copy the generated root filesystem/kernel Image to the current directory, and run the provided scripts to launch QEMU, e.g.:

1. `export QEMU_PATH=path/to/qemu/repository`
2. `cd meta-adlnx/qemu`
3. `cp ../../poky/build/tmp/deploy/images/admvpx39z/Image-initramfs-admvpx39z.bin ./`
4. `./run_qemu.sh`
5. `In a new shell, ./run_pmu.sh`

Extending and developing

Alpha Data's board support meta layer (**meta-adlnx**) is only a basic example. If your FPGA design requires custom drivers and modifications to the device tree, the meta layer can be modified to include these.

Revision History

Revision	Date	Description of Change
1.0	14th June 2018	Initial draft
1.1	1th Feburary 2019	Initial draft

Address: 4 West Silvermills Lane
Edinburgh, EH3 5BD, UK
Telephone: +44 131 558 2600
Fax: +44 131 558 2700
email: sales@alpha-data.com
website: http://www.alpha-data.com

Address: 611 Corporate Circle, Suite H
Golden, CO 80401
Telephone: (303) 954 8768
Fax: (866) 820 9956 - toll free
email: sales@alpha-data.com
website: http://www.alpha-data.com